An Optimal Broadcast Algorithm for Content-Addressable Networks

Ludovic Henrio, Fabrice Huet, and Justine Rochas

Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
ludovic.henrio@cnrs.fr,fabrice.huet@unice.fr,justine.rochas@inria.fr

Abstract. Structured peer-to-peer networks are powerful underlying structures for communication and storage systems in large-scale setting. In the context of the Content-Addressable Network (CAN), this paper addresses the following challenge: how to perform an efficient broadcast while the local view of the network is restricted to a set of neighbours? In existing approaches, either the broadcast is inefficient (there are duplicated messages) or it requires to maintain a particular structure among neighbours, e.g. a spanning tree. We define a new broadcast primitive for CAN that sends a minimum number of messages while covering the whole network, without any global knowledge. Currently, no other algorithm achieves those two goals in the context of CAN. In this sense, the contribution we propose in this paper is threefold. First, we provide an algorithm that sends exactly one message per recipient without building a global view of the network. Second, we prove the absence of duplicated messages and the coverage of the whole network when using this algorithm. Finally, we show the practical benefits of the algorithm throughout experiments.

Keywords: Broadcast, Peer-to-Peer, Content-Addressable Network

1 Introduction

In this work, we are interested in *Structured Overlay Networks* (SONs) where peers are organised in a well-defined topology and resources are stored at a deterministic location. The underlying geometric topology is used by communication primitives and ensures their efficiency. We are interested in CAN (Content-Addressable Network) [1] P2P networks, where peers are organised according to a multi-dimensionary cartesian space. This space is organised in a geometrical way; the geometrical organisation dictates the dependencies between peers, as we will see in Section 2.

This paper presents a broadcast algorithm for the CAN overlay network that prevents a peer from receiving the same message more than once. We call such a broadcast algorithm *efficient*, in the sense that it minimises the number of exchanged messages between peers. Of course, a broadcast algorithm also has to be *correct*, and reach every peer of the network.

In previous works, Bongiovanni and Henrio proved, using the Isabelle/HOL theorem prover, that an efficient broadcast protocol for CAN *existed* [2]. However, the algorithm that was exhibited to prove the *existence* of an optimal solution was naive and had a very high latency, making it unusable in practice. In this work, we are interested in the design and implementation of an effective broadcast protocol that, in practice, also has an acceptable latency.

The contribution of this paper is as follows:

- Firstly, we propose a new broadcast algorithm that greatly improves the state of the art.
- Secondly, we prove that this algorithm is both correct (it covers the whole network) and optimal in terms of exchanged messages.
- Thirdly, we set an experimental comparison of the algorithm with others in a realistic distributed environment and show its efficiency in practice.

This paper is organised as follows. First, Section 2 will show that several broadcast algorithms exist for CAN but none of them was able to completely remove duplicated messages purposes. Section 3 will present our broadcast algorithm, together with its proof of efficiency and correctness. Section 4 will present the evaluation of our algorithm over a distributed peer-to-peer network. Finally Section 5 concludes this paper.

2 Related Works and Objectives

2.1 Context and Motivation

A *CAN* [3] is a structured P2P network based on a d-dimensional Cartesian coordinate space labeled $\mathcal{D}$. This space is dynamically partitioned among all peers in the system such that each node is responsible for storing data, in the form of *(key, value)* pairs, in a sub-zone of $\mathcal{D}$. To store a (k, v) pair, the key k is deterministically mapped onto a point in $\mathcal{D}$ and the value v is stored by the node responsible for the zone comprising this point. The search for the value corresponding to a key k is achieved by applying the same deterministic function on k to find the node responsible for storing the corresponding value. These two mechanisms are performed by an iterative routing process starting at the query initiator and which traverses its adjacent neighbours (a peer only knows its neighbours), and so on and so forth until it reaches the zone responsible for the key to store/retrieve. One can find several definitions for a valid CAN, i.e. which shape can the zone of each peer have and how peers can be organised (see [2]). Here we rely on a very generic and simple definition: each zone is an hyperrectangle, and the only structure is the neighbouring relation: each peer only knows the peers whose zones are adjacent to its own zone. Additionally, a CAN is a torus, and the peers on the left border know the ones on the right border, but we will not use this feature in this paper. Figure 1 shows a 2-dimension CAN and some exchanged messages between neighbours.

Filali et al. [4] used a CAN to store large set of RDF data, and to perform queries taken from the BSBM benchmark [5]. They realised that the multicast

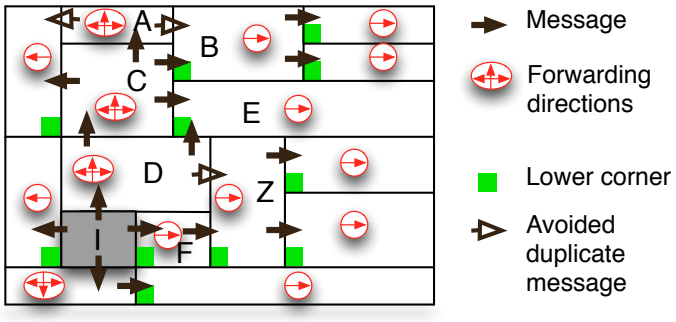

Fig. 1: M-CAN - Message forwarding

queries over several dimensions of the CAN did not scale properly because even the best performing broadcast algorithm generates a lot of duplicate messages (Section 4). These messages take valuable network resources, decreasing the overall performance. Our objective is to design an efficient broadcast algorithm that minimises the number of communications and that is only based on local information in a CAN.

2.2 Positioning

Problem statement The basic problem of optimal broadcast in a CAN is that, as a CAN is a P2P network, each peer only has information about the zone it manages, and the zones managed by its neighbours. Consequently, it is impossible to split the entire network into sub-spaces such that each zone exactly belongs to one sub-space: in Figure 1, the initiator has no knowledge about **Z** and cannot know that it must give the whole responsibility for zone **Z** to either **D** or **F**. Indeed, the initiator could decide that **F** is responsible for the lower half of **Z**, and that **D** is responsible for the upper half. In that case, **Z** would receive the message twice. It is possible to design an optimal algorithm based on sub-spaces, but this algorithm is inefficient because it almost never splits the space to be covered, and only one message is communicated at a time[1] [2]. Consequently, contrarily to the case of Chord [6], a broadcast algorithm for CAN that is both efficient and optimal cannot simply rely on the partitioning of the space to be covered.

Robustness One can argue that having duplicated messages should increase the robustness of the algorithm in case of failure, but there are much more efficient ways to replicate the messages than an inefficient algorithm. A much better way to ensure robustness would be simply to perform two efficient broadcasts carrying the same message from two different initiators and along different directions. In M-CAN [7], for example, some nodes receive the message once, while others can

[1] More precisely the space to be covered is only split if it is not path-connected.

receive it an arbitrarily high number of times, in an unpredictable manner. This is clearly not the best way to ensure robustness.

Churns Peers joining and leaving during a communication might require additional mechanisms to ensure that each peer correctly receives the message. Dealing with this issue generally relies on low-level synchronisations that depends on the implementation of CAN and is out of scope here. However, in order to tolerate churns between two broadcasts, our algorithm must rely only on the structure provided by CAN. For example, a classical additional structure for efficient broadcast is a *spanning tree* [8] but we do not use such a structure here because it is difficult and costly to maintain on an evolving CAN.

Multicast A crucial question is whether the primitive we aim for is a broadcast or a multicast, i.e., whether it can be targeted at only some of the nodes. In M-CAN [7], the authors suggest to reduce the problem of multicast to the one of broadcast on another (CAN) network. While this approach is valid here, we are interested in multicast over a range of values, i.e. along hyperrectangles included in the CAN. Indeed, considering our definition of a CAN (each node is responsible for a hyperrectangle zone), the intersection between an hyperrectangle to be covered and a CAN remains a CAN, thus our algorithm is still valid to multicast on a range of coordinates, or to cover only a certain number of dimensions.

An alternative definition of CAN [3] keeps track of the history of joining nodes, which forms a tree. Using this tree as a spanning tree has two disadvantages: first, this would limit the contribution to a subset of all possible CAN. Second, this tree would not allow to perform range multicast because the restriction of a CAN to an hyperrectangle leads to disconnected branches.

Our approach is the only one that allows efficient multicast over any particular zone of a CAN, without relying on additional structures. Our algorithm additionally features the following characteristics (1) It can perform either *broadcast or range multicast*. (2) It avoids duplicates, while replication is generally needed in a peer-to-peer network; but for reliability reasons it should be added above an efficient algorithm *in a controlled way*. (3) It tolerates *churns in between two executions* of the algorithm as it only relies on the CAN structure; dealing with churns during communications could only be done specifically for a particular implementation of the CAN.

2.3 Related works

A lot of work has been dedicated to broadcast and multicast on overlay networks. The availability of efficient algorithms depends mostly on the ability to build a spanning tree on the overlay. A tree-based system such as P-Grid [9] offers a natural support for broadcast. Others such as Chord [10], Tapestry [11] or Kademlia [12], can be seen as *k-ary trees*. Based on this observations, authors in [6] propose an efficient broadcast algorithm. Although this work is close to our own, it cannot be applied to CAN overlays, as building and maintaining a spanning tree is difficult and costly.

M-CAN [7] is an application-level multicast primitive which is almost efficient, but does not eliminate all duplicates if the space is not perfectly partitioned

(i.e. if the zones managed by the peers have not an equal size). The authors measured 3% of duplicates on a realistic example. In a publish/subscribe context, Meghdoot [13], built atop CAN, proposes a mechanism that totally avoids duplicates but requires the dissemination to originate from one corner of the zone to be covered. In general, finding the corner of the area to be covered would introduce a significant overhead (in terms of messages), resulting in an inefficient broadcast.

Compared to those approaches, our algorithm can originate from any node of the CAN and still avoid duplicates. In this sense, we position our algorithm as an improvement of M-CAN that completely eliminates duplicates. Below, we describe more precisely the dissemination algorithm proposed by M-CAN, which is the closest work to our approach.

2.4 M-CAN

In the following, the broadcast starts from one particular node, that we will call the *initiator*. A message is sent along a given dimension (from 1 to D, where D is the dimension of the CAN), and according to a given *direction* (which is either *ascending* if the coordinates along the considered dimension are increasing, or *descending* in the other case). It is only possible to forward the message to a node that is a neighbour along the considered dimension and direction.

The basic steps of the M-CAN algorithm are as follows:

1. The initiator sends the message to all of its neighbours.
2. A node receiving the message from a neighbour along dimension i in direction *dir* will forward the message to neighbours:
   - along dimensions $1 \ldots (i\text{-}1)$
   - along dimension i in direction *dir*.

Figure 1 shows a 2-dimensional CAN where initiator **Init**, starts a broadcast. In this figure, since node **B** has received a message from **C** along dimension 1, in the *ascending* direction, node **B** will forward it only on the *ascending* direction in dimension 1. Node **C**, on the other hand, has received the message along dimension 2, in the *ascending* direction. Thus it will forward the message in both directions along dimension 1, and only in the *ascending* direction along dimension 2. In Figure 1, the set of directions that each node is responsible for is pictured with red circled arrows.

This algorithm can lead to duplicated messages. For example, node **B** receives the same message from **C** and **A**. A deterministic condition is used to remove some of the duplicates: a node only forwards the message if it abuts the lowest corner of the neighbour it wants to forward to. This deterministic condition is called the *corner criteria*. The lowest corner is defined here as the corner which touches the propagation dimension and minimises the coordinates in all other dimensions. According to this *corner criteria*, node **A** will not forward the message to node **B** since **A** does not touch **B**'s lowest corner. However,

this only removes duplicates arising from the first dimension and cannot be applied in higher dimensions, otherwise the correctness of the broadcast could not be ensured. This is why some duplicates are still left with this algorithm. For example in Figure 1, node **E** receives the message twice.

3 Efficient Broadcast Algorithm

Our algorithm extends M-CAN, and remove duplicated messages that arise in dimensions higher than one. For this, we introduce a *spatial constraint* that allows us to always apply the *corner criteria*: we always propagate on the first dimension of a constrained sub-CAN.

3.1 Principles

The algorithm reasons on a set of nodes, where each node manages a rectangular zone. Considering a dimension i, the lower bound and the upper bound of the zone managed by node N are denoted $\mathrm{N}.LB[\mathrm{i}]$ and $\mathrm{N}.UB[\mathrm{i}]$. We denote by D the dimension of the CAN. Each message is sent according to a dimension (between 1 and D), and according to a direction (either *ascending* or *descending*).

Remember that the *corner criteria* prevents duplicates along the first dimension on which all the nodes forward. To prevent duplicates in the second dimension, we constraint the algorithm to only send the message to nodes belonging to a particular hyperplane in the CAN space. Each of the nodes belonging to the hyperplane will be responsible for propagating the message along the first dimension. We define the hyperplane as a set of fixed values in each dimension but the last one. These values are arbitrarily chosen in the zone of the initiator and, together, form what we call the *spatial constraint*. This *spatial constraint* is then an hyperplane of dimension $d-1$. The nodes belonging to this hyperplane form a sub-CAN of dimension $d-1$. So we can recursively apply our algorithm on this sub-CAN, with an hyperplane of dimension $d-2$ as *spatial constraint*; and so on. When the hyperplane becomes a line, no duplicate can arise when following the propagation direction if we send the message to the only one neighbour that contains the line in this direction.

Here is how the algorithm works. When a message is received along dimension k, it is forwarded to neighbours along dimensions $1..k-1$ in both directions, and along dimension k in only one direction (*ascending* or *descending*, identically to the reception). We then apply our additional condition: among the neighbours that are left, we send the message only to the ones that intersect the *spatial constraint* on dimensions $1..k-1$, and that satisfy the *corner criteria* on dimensions $k+1..d$. All dimensions but k are thus constrained either by the *spatial constraint* or by the *corner criteria*. We show that this ensures efficiency and correctness of the algorithm in Section 3.3.

Figure 2 illustrates the algorithm on the same configuration as Figure 1. In Figure 2, there is only one *spatial constraint* (on dimension 1) because the CAN only has two dimensions. In this case, it is set to the upper bound of the initiator

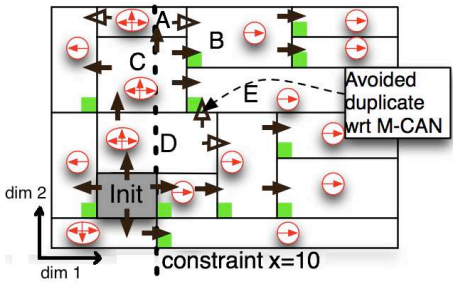

Fig. 2: Efficient broadcast in 2D

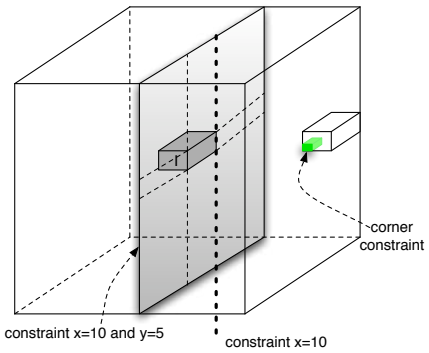

Fig. 3: Efficient broadcast in 3D

(node **Init**), where $constraintx = 10$. When **D** receives the message from **Init** along dimension 2, **D** only forwards the message to neighbours which intersect the line defined with $constraint\, x = 10$. Here, **D** only sends the message to **C**. **E** is also a neighbour of **D** along dimension 2 in the *ascending* direction, but **E** does not intersect the line. **E** will receive the message along dimension 1 afterwards. More formally, with a CAN of 2 dimensions, a node forwards the message to a neighbour if the following conditions are valid:

– when propagating along dimension 1:

$$Sender.LB[2] \leq Neighbor.LB[2] < Sender.UB[2]$$

– when propagating along dimension 2:

$$Neighbor.LB[1] \leq constraint[1] < Neighbor.UB[1]$$

As illustrated in Figure 3, this principle can be generalised to dimensions greater than 2. Thanks to our additional condition, we still have no duplicate. In dimension 3, the initiator first sends the message to the nodes intersecting a plane. In this plane, the problem is reduced to the example shown in Figure 2. In particular, one *spatial constraint* is used and a 2 dimensional *corner criteria* is applied. Then, when propagating along dimension 1, a three dimensional *corner criteria* is applied as depicted in Figure 3.

3.2 Broadcast Algorithm

We describe below the general algorithm in a more formal way. The data structures used in our algorithm are the following. A message embeds the *spatial constraint* that is transmitted without modification. The *spatial constraint* is a set of D coordinates that should represent a point belonging to the initiator node; for example it can be its lowest corner. constraint$[i]$ denotes the ith coordinate of this constraint. As the *spatial constraint* is transmitted without modification together with the message, we denote it as a global value. Each

message is sent and received along a given dimension ($dimension \in [1..D]$) and in a given direction ($direction \in \{ascending, descending\}$). Neighbours can be formally defined as follows:

Definition 1. *The neighbours of node N on dimension k and direction ascending are the set of nodes N' such that:*

$$N'.LB[k] = N.UB[k] \ \wedge \ \forall i \neq k.\ [N.LB[i], N.UB[i][\cap [N'.LB[i], N'.UB[i][\neq \emptyset$$

Symmetrically, neighbours of node N on dimension k and direction descending are the set of nodes N' such that:

$$N'.UB[k] = N.LB[k] \ \wedge \ \forall i \neq k.\ [N.LB[i], N.UB[i][\cap [N'.LB[i], N'.UB[i][\neq \emptyset$$

Algorithm 3.1 Efficient broadcast algorithm

```
1: upon event reception of message M on dimension d0 and direction dir0 on node
2:   for each k≤d0 do
3:     if k=D+1 then
4:       direction← ∅
5:     else
6:       if k < d0 then
7:         direction← {descending,ascending}
8:       else
9:         direction←dir0
10:    for each dir in direction do
11:      for each neighbour on dimension k and direction dir do
12:        for each i in 1 .. k − 1 do                    ▷ Spatial Constraint
13:          if not ( neighbour.LB[i] ≤ constraint[i] < neighbour.UB[i]) then
14:            skip neighbour
15:        for each i in k + 1 .. D do                    ▷ Corner Criteria
16:          if not( node.LB[i] ≤ neighbour.LB[i] < node.UB[i]) then
17:            skip neighbour
18:        send message on dimension k and direction dir to neighbour
19: end event
```

The detailed algorithm is given in Algorithm 3.1. Upon message reception along dimension $d0$, a node must forward it along lower dimensions (line 2) in both directions (line 7), and along dimension $d0$ in the same direction (line 9). For each neighbour in the considered dimensions and directions, their coordinates in dimensions lower than the propagating dimension are checked against the *spatial constraints* (line 12-14), and their coordinates in dimensions higher than the propagating dimensions are checked against the *corner criteria* (line 15-17). The *spatial constraint* condition on a dimension i checks that the neighbour's zone contains the ith value of the *spatial constraint* in the dimension i:

$$neighbour.LB[i] \leq \text{constraint}[i] < neighbour.UB[i]$$

The *corner criteria* on dimension i checks that, along dimension i, the lower bound of the neighbour in the dimension i is in the zone of the sender:

$$node.LB[i] \leq neighbour.LB[i] < node.UB[i]$$

If a neighbour verifies both conditions, the message is sent to it. This algorithm is initiated by sending a broadcast message to the initiator from an artificial dimension $D + 1$ (line 3).

3.3 Properties of the Algorithm

In the following, we prove the main properties of the algorithm. Those properties ensure that each node of the CAN receives the message exactly once. We first introduce two lemmas that are crucial to prove the properties of the algorithm.

Lemma 1 *If node N sends a message to node N' along dimension d and in direction dir then:*

$$\forall i < d.\, N'.LB[i] \leq constraint[i] < N'.UB[i]$$

and if N' is not the initiator (i.e., $d \leq D$) then:

- *either $dir = ascending$ and $N'.LB[d] > constraint[d]$,*
- *or $dir = descending$ and $N'.UB[d] \leq constraint[d]$.*

Proof. By recurrence on the length of the path needed to reach node N', i.e., on the number of messages needed to reach node N'.

The initiator artificially receives a message from outside the CAN on dimension $D + 1$; Here it is sufficient to verify:
$\forall i < D + 1.\, N'.LB[i] \leq \text{constraint}[i] < N'.UB[i]$
As the constraint must belong to the initiator node, this is trivial.

Now suppose that N' is not the initiator; node N sends a message to node N' on dimension d and from direction dir. First, as the message was sent from node N (possibly the initiator), by executing Algorithm 3.1, the algorithm ensures that $\forall i < d.\, N'.LB[i] \leq \text{constraint}[i] < N'.UB[i]$, else N' would have been skipped at line 16. Second, suppose $dir = ascending$ (the message is sent towards increasing coordinates). Then two cases are possible:

- N is the initiator and $d < D + 1$, then $N.LB[d] \leq \text{constraint}[d] < N.UB[d]$ (because the constraint belongs to the initiator's zone).
- N is not the initiator, thus there was a message sent from N_0 to N on dimension d' and direction dir'. By definition of the algorithm, we have two possibilities:
  - $d = d'$ and $dir' = ascending$; by recurrence hypothesis $N.LB[d] > \text{constraint}[d]$; additionally, we always have $N.UB[d] > N.LB[d]$.
  - $d < d'$; in that case, by recurrence hypothesis $N.LB[d] \leq \text{constraint}[d] < N.UB[d]$

In all cases, we have $N.UB[d] > \text{constraint}[d]$. As N' is a neighbour of N on dimension d and direction *ascending*, by Definition 1, $N'.LB[d] = N.UB[d]$, consequently, $N'.LB[d] > \text{constraint}[d]$.

The case where $dir = descending$ is similar: we have by recurrence $N.LB[d] \leq \text{constraint}[d]$, and by the neighbouring definition $N'.UB[d] \leq \text{constraint}[d]$.

The following corollary is a direct consequence of the preceding lemma.

Lemma 2 (Corollary) *If node N sends a message to node N' on dimension d and direction dir then:*

$$(\forall i < d'.\, N'.LB[i] \leq constraint[i] < N'.UB[i]) \Rightarrow d' \leq d$$

$$(N'.LB[i] > constraint[i] \vee N'.UB[i] \leq constraint[i]) \Rightarrow i \geq d$$

From the two lemmas above, we can prove the efficiency and correctness of the algorithm. First, our broadcast algorithm is efficient in the sense that the same message is never received twice by the same node:

Theorem 1 (Efficiency) *Two nodes cannot send the message to the same third one.*

Proof. We prove the theorem by contradiction: we suppose node N_1 sends the broadcast message on dimension d_1 and direction dir_1 to node N and that N_2 sends the broadcast message on dimension d_2 and direction dir_2 to node N, with $N_1 \neq N_2$.

Let us first prove that $d_1 = d_2$ by contradiction too. Suppose without loss of generality that $d_1 < d_2$, then by Lemma 1 applied on the message from N_2 to N on dimension d_2, as $d_1 < d_2$ we have $N.LB[d_1] \leq \text{constraint}[d_1] < N.UB[d_1]$. Additionally, by Lemma 1 applied on the message from N_1 to N we have either $dir = ascending$ and $N.LB[d_1] > \text{constraint}[d_1]$ or $dir = descending$ and $N.UB[d_1] \leq \text{constraint}[d_1]$. In both cases there is a contradiction; thus $d_1 = d_2$. Also $dir_1 = dir_2$, else the application of Lemma 1 would also lead to a contradiction.

Secondly, suppose again that $dir = ascending$ (the case *descending* is similar). By definition of Algorithm 3.1, the message was not skipped at line 21, neither by N_1 nor N_2, and so:
$\forall i \in d_1 + 1..D.\, N_1.LB[i] \leq N.LB[i] < N_1.UB[i]$
$\quad \wedge\ N_2.LB[i] \leq N.LB[i] < N_2.UB[i]$.

Additionally, as N is neighbour of N_1 and N_2 along dimension d_1 and direction *ascending*,
$N.LB[d_1] = N_1.UB[d_1] = N_2.UB[d_1]$ (Definition 1). Finally, we also have:
$\forall i \in 1..d_1 - 1.\, N_1.LB[i] \leq \text{constraint}[i] < N_1.UB[i]$
$\quad \wedge\ N_2.LB[i] \leq \text{constraint}[i] < N_2.UB[i]$,
because N_1 and N_2 themselves received the message on a dimension greater or equal to d_1 and by Lemma 1. Now consider the point P of coordinates:
$(\text{constraint}[1], .., \text{constraint}[d_1 - 1], N.LB[d_1] - \varepsilon, N.LB[d_1 + 1], .., N.LB[D])$
where ε is a small value (e.g., half the smallest dimension of the smallest zone of

the CAN). The arguments above allow us to prove that P is both in the zone of N_1 and in the zone of N_2, which is contradictory with the definition of a CAN: each point of the Cartesian space is managed by one and only one node. Hence N_1 and N_2 are necessarily the same node.

We proved that Algorithm 3.1 is efficient. Note that showing that the initiator does not receive the message twice needs a separate but similar proof. Finally, we can prove that this broadcast algorithm covers the whole network. Overall, we show that each node of the CAN receives the message exactly once.

Theorem 2 (Coverage) *Each node of the network receives the message.*

The proof of this theorem is provided in the research report associated to this paper [14]. It heavily relies on Lemma 1 and 2. The principle is simple: according to Lemma 1 and 2 we can deduce the node that is "responsible" for sending the message to each node. Consider the highest dimension on which a node does not intersect the constraint. If, on this dimension, this node is above the constraint, then the responsible node is a neighbour located along this dimension in the *descending* direction that should send it the message. Additionally, we say that a node N_1 is "closer" than another node N_2 to the constraint if either the highest dimension on which the node does not intersect the constraint is bigger for N_2 than for N_1 (i.e. N_1 meets more constraints), or if this dimension is the same, and the lower bound of N_1 is closer to the constraint than N_2 on this dimension. The proof works by contradiction: we consider N_0, the uncovered node the "closest" to the constraint. We then prove that the node N' "responsible" for sending the message to N_0 effectively meets all the constraints for sending the message and that it is "closer" to the constraint. If N' received the message it should have sent it to N_0, and if it did not, the N_0 is not the closest uncovered node as N' is closer. This is contradictory.

It is worth noticing that it is easy to make our algorithm robust to communication failures. Indeed, it is sufficient to perform two independent broadcasts from two different initiators, and reversing the role of each dimension, this way each node receives the message exactly twice and from different senders.

4 Evaluation

In this section we present experiments highlighting the performance of our algorithm. We show that, in realistic situations, it significantly reduces the volume of data exchanged. We have based our implementation on the EventCloud [15] platform. Entirely written in Java, EventCloud is a system that uses CANs as the underlying structure for event processing. It currently runs a flooding-based (naive) broadcast algorithm. We have added a version of our algorithm and an implementation of M-CAN to this framework, and studied the performance of these three algorithms.

4.1 Variation of the number of peers

Experimental setup We have experimented on a grid of four geographically distant clusters, using up to 200 physical machines. All the machines involved in the experiment have two 4-core CPUs and at least 16GB of memory. In each site, the machines are linked with a 1Gb/s Ethernet network. Inter-site communications rely on a 10Gb/s dark fiber.

The software setup was as follows. In all experiments we built CAN overlays with a variable number of peers (from 50 to 1500) and 5 dimensions. Applications that use CAN usually vary from two to an infinite number of dimensions, as in works [16, 17]. The improvement due to our algorithm is greater as the CAN has more dimensions, as detailed in [14]. We considered that 5 dimensions would be a good compromise to show that, even with a small number of dimensions, our algorithm can already achieve a meaningful speedup. Each peer runs in its own Java Virtual Machine and we ensure that no machine executes more than 8 peers. The construction of the overlay was performed using the canonical algorithm described in [1]: when a new peer wants to join the overlay, it randomly chooses a point in the whole space. It then finds the peer responsible for the zone where this point lies, and takes half of it.

Since we wanted our experiments to represent realistic scenarios and to compare the different algorithms in similar conditions, we have used the following experimental protocol:

1. A CAN is randomly built with a given number of peers.
2. For each algorithm, ten broadcasts are started simultaneously from different peers chosen at random.
3. Step 1 and step 2 are repeated ten times.

Experimental Results Figure 4 shows the average number of exchanged messages per broadcast algorithm. The horizontal lines highlight the optimal (minimum) number of messages required to cover the entire network. The naive broadcast algorithm produces a high number of duplicate messages. By contrast, the M-CAN algorithm improves a lot the naive algorithm but a non-negligible number of duplicate messages is still left, especially in large networks. With 1500 peers, 395 duplicate messages are recorded on average. Moreover, from the error bars, we can see that the M-CAN algorithm is unpredictable. The number of messages is very dependent on the CAN configuration and on the location of the broadcast initiator. This is why a particular execution can generate up to twice the optimal number of messages. On the other hand, our algorithm always requires the minimum number of messages in order to reach every peer in the network.

We have measured the total size of exchanged data for each algorithm. Note that the messages did not contain any useful payload, thus we only measure the cost of the broadcast operation. With 1500 peers in the network, the M-CAN algorithm generated 25.6 MB of data on average. Our algorithm generated only 20.3 MB of data on average, i.e. a 20% reduction. Aside, we have also experimented with various number of dimensions: from 2 to 15 [14]. The percentage of duplicate messages with 15 dimensions using M-CAN was 112%. But even with

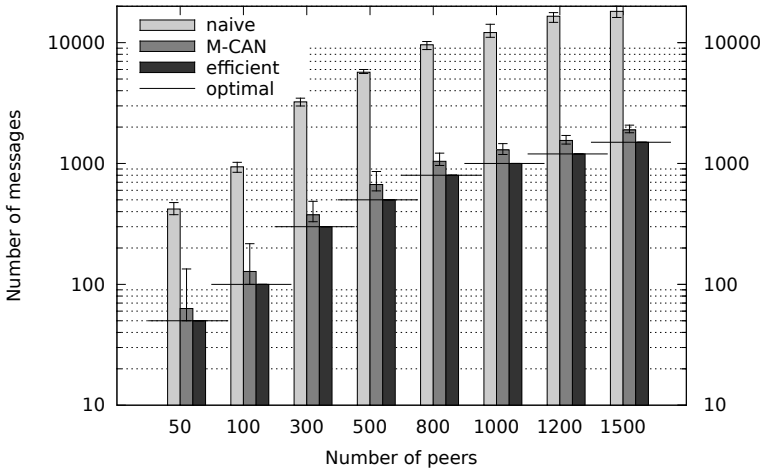

Fig. 4: Average number of messages and optimal number of messages with 5 dimensions

a small number of dimensions, from 3 to 5 dimensions, we measured in average 25% of duplicate messages with M-CAN. As a consequence, our algorithm is always significantly more efficient in terms of messages than M-CAN.

We have also measured the execution time, i.e. the time needed for each peer to receive at least one of the broadcast message. Figure 5 shows the average execution time of the three algorithms, and the speed up compared to the naive broadcast algorithm. The naive broadcast algorithm is significantly slowed down as the network grows. This is due to the quantity of duplicate messages that overload the network. On the other hand, both M-CAN and our algorithm maintain good performance as the network size increases. However, M-CAN exhibits a lower scalability because of the remaining duplicate messages. Compared to the naive broadcast on 1500 peers, M-CAN has only a speed-up of 5 whereas our algorithm reaches 8.

The previous experiments show that, although the number of duplicates with M-CAN is low, it still has a clear impact on realistic systems. Our algorithm, by totally avoiding duplicate messages, offers a significant improvement in terms of bandwidth and execution time, even when the CAN has a small number of dimensions.

5 Conclusion

In this article we have provided an algorithm for efficient broadcast over CAN peer-to-peer networks. We have proven that this algorithm covers the whole network, while preventing any node from receiving the same message twice. Moreover, it solely relies on the structure of the overlay and does not require to

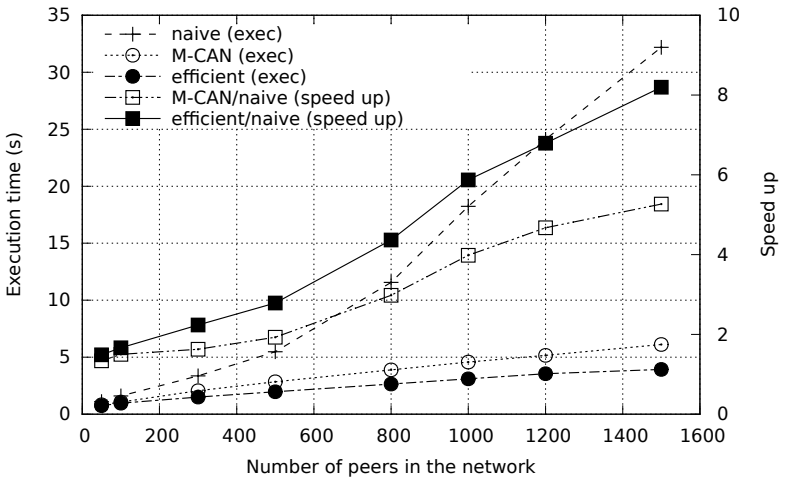

Fig. 5: Average execution time and speed up from the naive broadcast

maintain a spanning tree, which would be too costly. To show the practical usefulness of our algorithm, we have implemented it in a large scale platform and performed extensive experiments using up to 1500 peers on 200 physical machines. Our experiments show that the algorithm scales and completely prevents duplicated messages. Compared to the previously best broadcast algorithm, we reduce the amount of data on the network by up to 20%. As a consequence, when performing a high number of parallel broadcast queries, we were able to show a significant speedup compared to existing solutions.

Overall, this article shows that CAN overlays can be used effectively as information dissemination architectures. One of the main advantages of our approach is that we rely on a very broad definition of CAN overlays: a CAN is a N-dimensional space partitioned into hyperrectangles. As a consequence, our algorithm can be adapted to many variants of CAN, as long as zones are hyperrectangles and neighbours correspond to adjacent zones.

Acknowledgments This work was funded by the EU FP7 STREP PLAY (`www.play-project.eu`) and French ANR SocEDA (`www.soceda.org`). Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see `www.grid5000.fr`).

References

1. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proceedings of the 2001 Conference on Applications,

Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), ACM (2001) 161–172
2. Bongiovanni, F., Henrio, L.: A mechanized model for can protocols. In: 16th International Conference on Fundamental Approaches to Software Engineering (FASE'13). LNCS, Springer (2013)
3. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: A Scalable Content Addressable Network. NGC '01 Proceedings of the Third International COST264 Workshop on Networked Group Communications (August 2001)
4. Filali, I., Pellegrino, L., Bongiovanni, F., Huet, F., Baude, F., et al.: Modular p2p-based approach for rdf data storage and retrieval. In: Advances in P2P Systems. (2011)
5. Bizer, C., Schultz, A.: The berlin sparql benchmark. International Journal On Semantic Web and Information Systems (2009)
6. El-Ansary, S., Alima, L., Brand, P., Haridi, S.: Efficient broadcast in structured P2P networks. In: Peer-to-Peer Systems II. Springer (2003) 304–314
7. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-Level multicast using Content-Addressable Networks. NGC '01 Proceedings of the Third International COST264 Workshop on Networked Group Communications (2001)
8. Perlman, R.: An algorithm for distributed computation of a spanningtree in an extended lan. In: Proceedings of the ninth symposium on Data communications. SIGCOMM '85, ACM (1985)
9. Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-Grid: a self-organizing structured P2P system. ACM SIGMOD Record **32**(3) (2003) 33
10. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), New York, NY, USA, ACM (2001) 149–160
11. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: a resilient global-scale overlay for service deployment. Selected Areas in Communications, IEEE Journal on **22**(1) (2004) 41–53
12. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. IPTPS '01, London, UK, UK, Springer-Verlag (2002) 53–65
13. Gupta, A., Sahin, O., Agrawal, D., Abbadi, A.: Meghdoot: content-based publish/subscribe over P2P networks. In: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Springer-Verlag New York, Inc. (2004) 254–273
14. Henrio, L., Huet, F., Rochas, J.: An Optimal Broadcast Algorithm for Content-Addressable Networks – Extended Version. Research report (September 2013) `http://hal.inria.fr/hal-00866228`.
15. INRIA: The EventCloud middleware (2012) `http://www.play-project.eu/solutions/event-cloud`.
16. Li, M., Ye, F., Kim, M., Chen, H., Lei, H.: Bluedove: A scalable and elastic publish/subscribe service. IPDPS (2011)
17. Anceaume, E., Le Merrer, E., Ludinard, R., Sericola, B., Straub, G.: Fixme: A self-organizing isolated anomaly detection architecture for large scale distributed systems. OPODIS (2012)